Adaptive Testing of Compute-in-Memory Based CNNs Using Probabilistic Test Acceptance Limits

Anurup Saha[1], Kwondo Ma[2], Chandramouli Amarnath[3], Moinuddin Qureshi[1] and Abhijit Chatterjee[1]
[1]School of Electrical and Computer Engineering, Georgia Institute of Technology, [2]Rebellions, South Korea, [3]Google, USA
Email: asaha74@gatech.edu, magunudo@gmail.com, chandamarnath@google.com,
moin@gatech.edu, abhijit.chatterjee@ece.gatech.edu

2025 IEEE 31st International Symposium on On-Line Testing and Robust System Design (IOLTS) | 979-8-3315-3334-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/IOLTS65288.2025.11117076

***Abstract*—Compute-in-memory (CiM) based convolutional neural network (CNN) accelerators achieve low-power inference, utilizing memristive crossbar arrays for matrix multiplications. However, inherent conductance variations within the crossbar introduce computational errors. These errors propagate to the CNN output and cause image misclassification, leading to substantial accuracy degradation. This paper addresses the critical challenge of efficient and reliable post-manufacture testing for CiM-based CNN accelerators. We propose a novel test image sampling methodology, which iteratively applies sampled images from the CNN's testing dataset using progressive random sampling (PRS) to a device under test (DUT) and estimates a confidence interval for the DUT accuracy. Based on the confidence interval and the acceptable accuracy threshold, the test labels a DUT as "pass" or "fail". Furthermore, if we have access to an initial set of DUTs, we apply the images from the CNN's testing dataset to these DUTs and leverage the DUT outputs to rank-order test images. We develop a sequential estimation test (SET) framework, where the images from the CNN's testing dataset are sequentially applied according to a predetermined rank and the test terminates when a DUT can be confidently labeled as "pass" or "fail" based on the applied images. In each case, the number of applied test images *adapts* to the quality of the DUT. Experiments show that PRS and SET achieve $2.2\times$ and $4.6\times$ speedup compared to state-of-the-art test methodologies.**

***Index Terms*—Compute-in-memory, Convolutional neural network, Adaptive testing.**

I. Introduction

The increasing growth of deep learning applications has led to demand for energy-efficient hardware accelerators. Analog compute-in-memory (CiM) based architectures eliminate the memory-wall bottleneck of traditional Von-Neumann architectures by removing separation of processing elements and memory [1]. In these architectures, memristive crossbar arrays (MCA) store the weights of a CNN and perform in-place matrix multiplication using Kirchhoff's current law. Emerging memory technologies such as resistive random access memory (RRAM) and magnetoresistive random access memory (MRAM) are commonly used to implement memristive crossbar arrays [2]. The works of [3], [4] have explored how CNNs can be accelerated using CiM architectures.

Despite these advantages, analog CiM architectures face several challenges that hinder their widespread adoption. Unlike conventional digital circuits, these systems operate using analog computation principles, making them more susceptible to noise and non-idealities in memristive cells. As a result memristor-based implementations of CNNs achieve lower classification accuracy compared to their digital counterparts. Further, the conductance variation profile of every manufactured crossbar is unique. These challenges demand robust testing strategies for every manufactured crossbar-based CNN to ensure reliable CiM-based deep learning acceleration.

Adaptive testing of analog circuits has been explored in [5], [6]. Similarly, testing of AI accelerators has been widely studied [7]. The work of [8] uses machine learning methods to evaluate fault criticality in machine learning accelerators. In [9], test pattern generation to detect functional safety violations is developed. In [10], a compact functional test for DNNs is proposed. A machine learning assisted alternate test architecture is proposed in [11]. Among these methods, alternate test achieves state-of-the-art test accuracy as well as significant speedup compared to exhaustive testing. Alternate test has also been successfully applied to spiking neural networks in [12]. Despite the benefits, alternate test suffers from two major limitations: (1) It requires training a regressor using DUT output measurements. (2) Due to prediction error of the regressor, the speedup can be limited in certain scenarios.

In this research, a novel CiM-based CNN test methodology is developed that is suitable for test environments where the *entire test image dataset* (numbering 10,000 for CIFAR-10) is available for testing each DUT during manufacturing test. Consequently, training a regressor for predicting DUT classification accuracy (expensive, requires access to the outputs of neurons in the CNN layers) is not a test requirement. Moreover, test decisions are based solely on whether a selected test image is classified correctly or not. So no access to neuron outputs as in [11], is needed. Estimates of DUT accuracy are computed after every applied test image derived from *progressive random sampling (PRS)* of test images and the uncertainty in estimating the DUT accuracy is bounded using statistical methods. Based on the estimated accuracy, the uncertainty bounds and the accuracy threshold, either the DUT is labeled as "pass" or "fail" or the next image is applied to reduce the uncertainty in estimating DUT accuracy. The technique *adapts to the quality of the DUT (CNN)*. If the DUT accuracy is significantly higher (or lower) than the acceptable classification accuracy of the DUT, fewer test images are needed (this identifies devices that are clearly "good" or "bad", respectively). Marginal DUTs are tested more aggressively.

To speed up the testing progress derived from PRS above, we propose a sequential estimation test (SET) procedure, which applies test images in a deterministic order. The images

979-8-3315-3334-2/25/$31.00 ©2025 IEEE

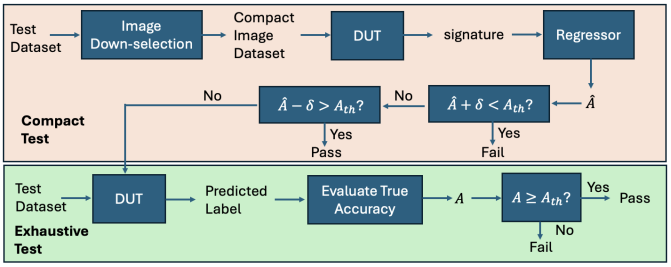

Figure 1: Overview of Alternate Test

are ordered based on their ability to accurately estimate accuracies of DUTs with a *specified confidence of estimation* using a greedy algorithm. After every applied image, the maximum and minimum probable normalized accuracies for each DUT are calculated and the test terminates when it can categorize a DUT as "pass" or "fail". In summary this research makes the following contributions:

(1) We propose a progressive random sampling based test image selection approach for testing CiM based CNN accelerators. This does not require use of a regressor and is more efficient than state of the art techniques in terms of test efficiency.

(2) We develop a sequential estimation test (SET) framework, which rank orders test images in a greedy manner based on error minimization. SET achieves $4.6\times$ speedup compared to state-of-the-art alternate testing methods [11].

II. Prior Work and Limitations

A. Preliminaries of Regressor Based Alternate Test

Assume that a CiM based CNN (also referred to as device under test or DUT) is evaluated using an image classification dataset $\mathscr{T} = \{im_1, im_2, \cdots im_T\}$. Let α_t denote whether the t-th image is correctly classified by the DUT ("1" for correct classification and "0" for misclassification). The normalized accuracy (p) and the accuracy (A) of the DUT are defined as:

$$p = \frac{1}{T}\sum_{t=1}^{T}\alpha_t \tag{1}$$

$$A = p \times 100\% \tag{2}$$

If the acceptable accuracy threshold is A_{th}, the goal of the test framework is to categorize a DUT as "pass" ($A \geq A_{th}$) or "fail" ($A < A_{th}$). From Equation (2), the accuracy of a DUT can be calculated by applying all T images to a DUT and calculating all α_t values. Such a method is referred as *exhaustive test*.

Exhaustive test incurs high test time as it requires applying T images to every DUT. The key motivation of regressor-based alternate test, as explained in Figure 1, is to label a DUT as "pass" or "fail" *without having to apply all T images*. In this framework, an image down-selection algorithm is used to create a compact image dataset $\mathscr{C}$ with K images ($K << T$). In [11], image down-selection is performed by hierarchical clustering of the binary classification results of each image across a set of DUTs selected from diverse process corners. The DUT logits (final layer outputs) corresponding to application of each image from $\mathscr{C}$ are monitored. The logits corresponding to all images are concatenated to extract a DUT signature. The signature is passed to a trained regressor which predicts the accuracy of the DUT as $\hat{A}$. If the maximum prediction error due to finite expressive power of a regressor is δ, the regressor prediction is used to classify the DUT if $|A_{th} - \hat{A}| > \delta$. Otherwise the DUT undergoes exhaustive testing. During exhaustive testing, all images from $\mathscr{T}$ are applied to the DUT and the DUT accuracy is calculated as the percentage of correctly classified images. Based on the calculated accuracy, a DUT is labeled as "pass" or "fail".

B. Limitations of Alternate Test

There are two key limitations of the alternate test approach [11] for CiM based CNNs. First, this requires the use of a trained regressor to predict the DUT (CNN) classification accuracy from the DUT response signature to the applied test images in $\mathscr{C}$. To train the regressor, an exhaustive test image set is applied to D DUTs from diverse process corners to obtain their classification accuracies. Also, the response signature of each DUT to the applied compact image set $\mathscr{C}$ is extracted. For each DUT, this results in a response signature sig_d and an corresponding classification accuracy A_d. The data $\{(sig_1, A_1), (sig_2, A_2), \cdots, (sig_D, A_D)\}$ is used to train the regressor. Second, the alternate test approach can suffer from limited test speedup which is defined as the ratio of the number of images in $\mathscr{T}$ and the average number of images which are applied to a DUT in alternate test framework. Assume that, the fraction of DUTs requiring exhaustive testing is β. Then $(1-\beta)$ fraction of DUTs are tested with K images (using regressor prediction) whereas β fraction of the DUTs require applying T images. The average number of images required to test a DUT is $\beta T + (1-\beta)K = K + \beta(T-K) \approx K + \beta T$ ($K << T$). The speedup is:

$$\eta = \frac{T}{K + \beta T} = \frac{1}{\beta + \frac{K}{T}}$$

If the regressor used in alternate test has exact prediction capability, then $\delta \rightarrow 0$ and $\beta \rightarrow 0$. In such scenario, speedup of T/K is achieved. However due to finite expressive power of a regressor ($\delta > 0$), in many scenarios we observe, $\beta >> K/T$. In such cases, the speedup is merely $1/\beta$, where $1/\beta << T/K$.

III. Crossbar Variability Model

Convolutional neural networks consist of convolution, linear, max-pool, batch-normalization and rectified linear unit (ReLU) activation layers. Among these layers, linear layers use matrix vector multiplication (MVM) during inference. On the other hand, convolution layers require general matrix multiplication (GeMM) which can further be represented as a sequence of MVMs. Memristive crossbar arrays can efficiently compute MVM where the stationary weight matrix is stored within a crossbar. Each weight is stored using a memristive device of equivalent conductance. The inputs to the crossbar are generated in the form of analog voltages using digital-to-analog converters (DACs). The current outputs from the crossbar are converted back to real numbers using analog-to-digital converters (ADCs). The ADC outputs constitute the MVM outputs and are passed to digital computation units for maxpool and batch-normalization and activation operations.

We refer the reader to [4] for more details about CNN acceleration using compute-in-memory architectures.

MVM outputs from an analog crossbar suffer from computational errors due to stochastic conductance variations of memristive cells, which manifest in the form of device-to-device and cycle-to-cycle conductance variations [13]. Inter-chip device-to-device variations impact all memristor cells in a crossbar equally. Intra-chip device-to-device variation has spatially correlated and independent random components [14]. Cycle-to-cycle variations can be modeled as independent random variation. As a result, we model conductance variation as a sum of systematic, spatially correlated and independent random variations. The goal of the variability model is to perform CNN inference such that the impact of stochastic conductance variation on image classification can be modeled accurately. To do so, we replace each weight of the convolution and dense layers of the CNN with corresponding non-ideal weight values. We modify the i-th weight of a layer as:

$$w_i^{'} = w_i \times (1 + \epsilon_i) = w_i \times (1 + \epsilon^s + \epsilon_i^c + \epsilon_i^r) \tag{3}$$

Here ϵ^s, ϵ_i^c, and ϵ_i^r refer to systematic, spatially correlated and random non-ideality factors. We refer to ϵ_i as the non-ideality factor. Following prior work [15], for each DUT we sample a shared systematic non-ideality factor ϵ^s from a zero mean normal distribution with variance σ_{sys}^2. On the other hand, the random non-ideality factor corresponding to each weight is independently sampled from another zero mean normal distribution with variance σ_{rand}^2. The spatially correlated non-ideality factor is sampled from a spacial covariance matrix Γ, where the (i, j)-th element of Γ represent the spatial correlation between the memristive cells storing the i-th and j-th weight. We refer the reader to [14], [15] for calculation of the spatial covariance matrix.

IV. Progressive Random Sampling

The core idea of progressive random sampling (PRS) is to *iteratively apply a set of random images* from the CNN's testing dataset to a DUT (starting with a single image), estimate its normalized accuracy based on the applied images and compute a confidence interval around the estimated normalized accuracy of the DUT. Based on the confidence interval and acceptable accuracy threshold, PRS checks whether it is possible to label a DUT as "pass" or "fail" with high confidence (specified). If this is not possible, the test image set is expanded with another randomly selected image and the process is repeated till a DUT can be labeled as "pass" or "fail" with high confidence. Figure 2 illustrates PRS with an example. Here p_{th} is the acceptable normalized accuracy threshold. In the first iteration im_1 is applied to the DUT and the confidence interval of the estimated normalized accuracy is $[p_{lo}, p_{hi}]$. Since p_{th} is within $[p_{lo}, p_{hi}]$ it is not possible to label the DUT as "pass" or "fail" In the following iterations im_4, im_3 and im_2 are applied to the DUT. As more images are applied, the range of the confidence interval shrinks and after the fourth iteration the DUT can be labeled as "fail".

The rest of this section is organized as: (1) We first define the estimated normalized accuracy of a DUT $\hat{p}_t$ when t randomly selected images are applied to the DUT (Equation (4)). (2) We derive a confidence interval $[p_{lo}, p_{hi}]$ such that the true normalized accuracy of the DUT $p \in [p_{lo}, p_{hi}]$ with high probability (p is defined in Equation (1)). (3) We use the calculated confidence interval to develop a testing algorithm (Algorithm 1).

A. Estimated Normalized Accuracy and Confidence Interval

Consider a DUT being evaluated using the testing dataset of the CNN $\mathscr{T}$ consisting of T images $\mathscr{T} = \{im_1, im_2, \cdots, im_T\}$. Equation (1) defines the normalized accuracy of the DUT. Our goal is to accurately estimate p using t images ($t \leq T$). If we randomly pick t images from $\mathscr{T}$, and α_k is a random variable representing whether the k-th image is correctly classified ($1 \leq k \leq t$, $\alpha_k = 1$ for correct classification and 0 for misclassification), then α_k is a Bernoulli random variable with mean p. We define the estimated normalized accuracy (ENA) as:

$$\hat{p}_t = \frac{1}{t}\sum_{k=1}^{t} \alpha_k \tag{4}$$

Next, we want to calculate p_{lo} and p_{hi} such that $P(p_{lo} \leq p \leq p_{hi}) \rightarrow 1$. To derive the confidence interval, we first calculate the mean and variance of $\hat{p}_t$. Since $\hat{p}_t$ is an average of independent Bernoulli variables, the central limit theorem implies that for a sufficiently large $t \geq T_{min}$, $\hat{p}_t$ follows a normal distribution with mean p and variance $\sigma_t^2 = \frac{p(1-p)}{t}$.

We provide a formal proof of the mean and variance of $\hat{p}_t$ using the fact that α_k is a Bernoulli random variable and $\mathbb{E}[\alpha_k] = p$. We calculate the mean of $\hat{p}_t$ as:

$$\mathbb{E}[\hat{p}_t] = \mathbb{E}[\frac{1}{t}(\sum_{k=1}^{t} \alpha_k)] = \frac{1}{t}\sum_{k=1}^{t} \mathbb{E}[\alpha_k] = \frac{1}{t}\sum_{k=1}^{t} p = p$$

The variance of $\hat{p}_t$ can be calculated as:

$$var(\hat{p}_t) = \mathbb{E}\big[\hat{p}_t^2\big] - \mathbb{E}\big[\hat{p}_t\big]^2 = \mathbb{E}\big[(\frac{1}{t}\sum_{k=1}^{t} \alpha_k)^2\big] - p^2$$
$$= \frac{1}{t^2}\mathbb{E}\big[\sum_{i=1}^{t}\sum_{j=1}^{t} \alpha_i\alpha_j\big] - p^2$$

Now, if $i \neq j$, then α_i and α_j are independent. As a result,

$$\mathbb{E}[\alpha_i\alpha_j] = \mathbb{E}[\alpha_i] \times \mathbb{E}[\alpha_j] = p \times p = p^2$$

If $i = j$, we obtain:

$$\mathbb{E}[\alpha_i\alpha_j] = \mathbb{E}[\alpha_i^2] = P(\alpha_i = 0) \times 0 + P(\alpha_i = 1) \times 1 = p$$

Using the value of $\mathbb{E}[\alpha_i\alpha_j]$,

$$var(\hat{p}_t) = \frac{1}{t^2}\mathbb{E}\big[\sum_{i=1}^{t}\sum_{j=1}^{t} \alpha_i\alpha_j\big] - p^2$$
$$= \frac{1}{t^2}\mathbb{E}\big[\sum_{i=1}^{t}\sum_{j=1, j\neq i}^{t} \alpha_i\alpha_j + \sum_{i=1}^{t} \alpha_i^2\big] - p^2$$
$$= \frac{1}{t^2}\big((t^2 - t)p^2 + tp\big) - p^2$$
$$= \frac{1}{t}p(1 - p)$$

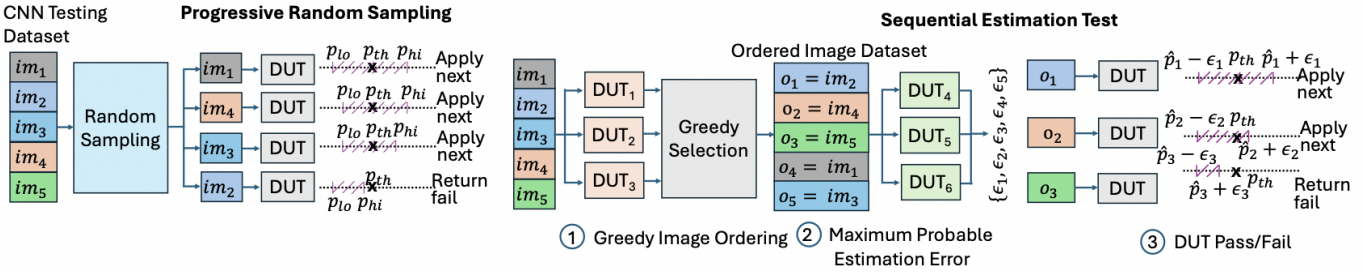

Figure 2: Overview of Progressive Random Sampling and Sequential Estimation Test

Based on the mean and variance of $\hat{p}_t$, we calculate p_{lo} and p_{hi} such that $P(p_{lo} \leq p \leq p_{hi}) \rightarrow 1$ We define the approximate lower bound and approximate upper bound of the estimated normalized accuracy as:

$$p_{lo} = \hat{p}_t - m_{prs}\sigma_t \tag{5}$$

$$p_{hi} = \hat{p}_t + m_{prs}\sigma_t \tag{6}$$

Here m_{prs} refers to the confidence multiplier for PRS. Reliable testing using progressive random sampling requires the true normalized accuracy of a DUT to be within the confidence interval $[p_{lo}, p_{hi}]$. The probability that $p \in [p_{lo}, p_{hi}]$ can be calculated as:

$$\begin{aligned} &P(p_{lo} \leq p \leq p_{hi}) \\ &= P(\hat{p}_t - m_{prs}\sigma_t \leq p \leq \hat{p}_t + m_{prs}\sigma_t) \\ &= P(|\hat{p}_t - p| \leq m_{prs}\sigma_t) \\ &= P(p - m_{prs}\sigma_t \leq \hat{p}_t \leq p + m_{prs}\sigma_t) \qquad (7) \\ &= erf(m_{prs}/\sqrt{2}) \qquad (8) \end{aligned}$$

By definition of cumulative density function (CDF) of a normal distribution, Equation (7) reduces to Equation (8). Here $erf(x)$ refers to the error function which is defined as $erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$.

For example, for $m_{prs} = 1.96$ and 3 we obtain $P(p_{lo} \leq p \leq p_{hi}) = 0.95$ and 0.997 respectively. However, p_{lo} and p_{hi} cannot be exactly calculated as σ_t depends on p, which is an unknown DUT-dependent quantity. For estimation purposes, we approximate $\sigma_t \approx \sqrt{\frac{1}{t}\hat{p}_t(1-\hat{p}_t)}$. In Section IV-B, we use p_{lo} and p_{hi} to develop a testing framework.

B. Test Framework

Algorithm 1 explains the overall testing framework. It starts with a DUT and an image classification dataset $\mathscr{T}$. At every iteration of the test, a randomly sampled image is applied to the DUT and based on DUT output α_t is set to '1' or '0' (line 7). Next, we estimate the normalized accuracy $\hat{p}_t$ and derive approximate lower and upper bounds for the estimated normalized accuracy p_{lo}, p_{hi} (line 9-11). If p_{th} lies outside the interval $[p_{lo}, p_{hi}]$, based on p_{lo} and p_{hi}, we classify the DUT as "pass" or "fail" (line 12 - 18). Otherwise, we keep repeating the steps with a new image.

Algorithm 1 Progressive Random Sampling

1: **Input:** Image dataset $\mathscr{T} = \{im_1, im_2, \cdots, im_T\}$, accuracy threshold A_{th}, normalized accuracy threshold $p_{th} = \frac{A_{th}}{100}$, device under test (DUT). confidence multiplier m_{prs}
2: **Initialize:** $\mathscr{X} = \{\}$
3: **for** $t = 1$ **to** T **do**
4: Randomly sample an image im_t from $\mathscr{T} - \mathscr{X}$
5: Add im_t to $\mathscr{X}$
6: Apply im_t to DUT
7: Set $\alpha_t = \begin{cases} 1 & \text{if } im_t \text{ is correctly classified} \\ 0 & \text{otherwise} \end{cases}$
8: Estimated normalized accuracy: $\hat{p}_t = \frac{1}{t}\sum_{k=1}^{t} \alpha_k$
9: Compute the standard deviation:
10: $\sigma_t = \sqrt{\frac{\hat{p}_t(1-\hat{p}_t)}{t}}$
11: Compute approximate lower and upper bounds of the estimated normalized accuracy
$p_{lo} = \hat{p}_t - m_{prs}\sigma_t$ and $p_{hi} = \hat{p}_t + m_{prs}\sigma_t$
12: **if** $p_{lo} > p_{th}$ and $t \geq T_{min}$ **then**
13: **Return** *pass*
14: **end if**
15: **if** $p_{hi} < p_{th}$ and $t \geq T_{min}$ **then**
16: **Return** *fail*
17: **end if**
18: **end for**
19: **if** $\hat{p}_T \geq p_{th}$ **then**
20: **Return** *pass*
21: **else**
22: **Return** *fail*
23: **end if**

V. Sequential Estimation Test

In progressive random sampling, the test images are applied in random order. If we have access to an initial set of DUTs to design a test framework, the statistics of the DUT outputs corresponding to the CNN testing dataset can be used to create a *deterministic order in which images should be applied during testing to maximize test speedup*. Figure 2 shows the three steps of the proposed sequential estimation test (SET):
(1) We apply the CNN testing dataset images to a set of DUTs (DUTs 1, 2 and 3 in Figure 2). Based on the DUT outputs we use a greedy algorithm to rank order the images and create an ordered image set. During DUT testing, the images are sequentially applied based on this determined order.
(2) We apply the first t images from the ordered image set to a set of DUTs (DUTs 4, 5 and 6 in Figure 2) and calculate the difference between the true and estimated normalized classification accuracy of the DUTs. The maximum difference is defined as the *maximum probable estimation error* and is calculated for $1 \leq t \leq T$.
(3) During real-time DUT testing, images from the ordered image set are applied sequentially in determined order and the test termination criterion depends in the maximum probable accuracy estimation error (Algorithm 3).

A. Greedy Image Ordering

Greedy image ordering aims at minimizing the difference between estimated and true normalized accuracies of DUTs with as few images as possible. Algorithm 2 outlines the overall image ordering methodology. The algorithm starts with D_1 DUTs, where the normalized accuracy of the d-th DUT p^d is known. The ordered image set $\mathscr{O}$ is initialized as an empty set and the estimated normalized accuracy for each DUT $\hat{p}_0^d$ is initialed to 0. At every iteration we sample J images $\{x_j\}_{j=1}^{J}$ from $\mathscr{T} - \mathscr{O}$ (line 4). Each sampled image is applied to the D_1 DUTs and estimated normalized accuracy is calculated for each DUT after applying a sampled image x_j (line 6-8). For each sampled image, mean absolute error is calculated based on the difference of estimated and true normalized accuracies (line 10). The image which achieves the least error is appended to the ordered image set and the estimated normalized accuracy after applying t images is updated accordingly (line 12-14).

Algorithm 2 Greedy Image Ordering

1: **Input:** (1) D_1 DUTs, where the normalized accuracy of the d-th DUT is p^d (2) CNN testing dataset $\mathscr{T} = \{im_1, im_2, \cdots, im_T\}$
2: **Initialize:** (1) Ordered image set $\mathscr{O} = \{\}$ (2) Estimated normalized accuracy $\hat{p}_t^d = 0$ for $1 \leq d \leq D_1$.
3: **for** $t = 1$ **to** T **do**
4: Sample J images $\{x_1, \cdots x_J\}$ from $\mathscr{T} - \mathscr{O}$ where $J = \min\{|\mathscr{T} - \mathscr{O}|, J_{min}\}$
5: **for** each x_j **do**
6: **for** $d = 1$ up to D_1 **do**
7: Set $\alpha_j^d = \begin{cases} 1 & \text{if } d-\text{th DUT correctly classifies } x_j \\ 0 & \text{otherwise} \end{cases}$
8: Estimated normalized accuracy $\hat{p}_j^d = \frac{1}{t}[(t-1)\hat{p}_{t-1}^d + \alpha_j^d]$
9: **end for**
10: Calculate mean absolute error $e_j = \frac{1}{D_1}\sum_{d=1}^{D_1} |\hat{p}_j^d - p^d|$
11: **end for**
12: Set $o_t = x_{j'}$ where $j' = \arg\min_j e_j$
13: Set $\mathscr{O} = \mathscr{O} \cup o_t$
14: Update $\hat{p}_t^d = \frac{1}{t}[(t-1)\hat{p}_{t-1}^d + \alpha_{j'}^d]$ (for $1 \leq d \leq D_1$)
15: **end for**
16: **Return** $\mathscr{O}$

B. DUT Labeling

Assume that the normalized accuracy of a DUT is p^d and the estimated normalized accuracy using the first t images from $\mathscr{O}$ is $\hat{p}_t^d$. We measure $\hat{p}_t^d$ for D_2 DUTs (these DUTs are different from the previous D_1 DUTs). We define the maximum probable estimation error corresponding to t images as:

$$\epsilon_t = \max_{d \in \{1,2,\cdots,D_2\}} |p^d - \hat{p}_t^d| \tag{9}$$

Maximum probable estimation error is calculated for $1 \leq t \leq T$. Finally, using the ordered image set and the maximum probable estimation error, we develop the test framework, as outlined in Algorithm 3. We iteratively apply images from the ordered test set $\mathscr{O}$ and calculate the estimated normalized accuracy (line 3-5). Based on the maximum probable estimation error, we calculate minimum and maximum probable normalized accuracies (line 6). Here m_{set} refers to the confidence multiplier for sequential estimation test. If the minimum probable normalized accuracy is higher than p_{th} or the maximum probable normalized accuracy is lower than p_{th}, then DUT is labeled as "pass" or "fail" and the test terminates (line 7-12). Otherwise, the next image is applied and the process repeats. When all images in $\mathscr{O}$ are exhausted, the DUT is classified based on the estimated normalized accuracy and the accuracy threshold.

Algorithm 3 DUT Labeling

1: **Input** (1) Ordered image set $\mathscr{O} = \{o_t\}_{t=1}^{T}$ (2) Maximum probable estimation error $\{\epsilon_t\}_{t=1}^{T}$ and (3) a DUT (4) Accuracy threshold A_{th}, normalized accuracy threshold $p_{th} = \frac{A_{th}}{100}$
2: **for** $t = 1$ up to T **do**
3: Apply o_t to the DUT
4: Set $\alpha_t = \begin{cases} 1 & \text{if } o_t \text{ is correctly classified} \\ 0 & \text{otherwise} \end{cases}$
5: Estimated normalized accuracy $\hat{p}_t = \frac{1}{t}\sum_{k=1}^{t} \alpha_k$
6: Calculate (1) minimum probable normalized accuracy = $\hat{p}_t - m_{set}\epsilon_t$ (2) maximum probable normalized accuracy = $\hat{p}_t + m_{set}\epsilon_t$
7: **if** $\hat{p}_t + m_{set}\epsilon_t < p_{th}$ **then**
8: **Return** fail
9: **end if**
10: **if** $\hat{p}_t - m_{set}\epsilon_t > p_{th}$ **then**
11: **Return** pass
12: **end if**
13: **end for**
14: **if** $\hat{p}_T \geq p_{th}$ **then**
15: **Return** pass
16: **end if**
17: **Return** fail

VI. Experimental Results

A. Experimental Setup

We evaluate the proposed framework for VGG16 [16] and Mobilenet [17] architectures trained on the CIFAR-10 dataset [18]. With ideal weights, the classification accuracies of VGG16 and Mobilenet are 93.24% and 91.74% respectively. For variability modeling, if the variance of ϵ_i is σ_{tot}^2, we define $\frac{\sigma_{sys}^2}{\sigma_{tot}^2} \times 100\%$ as the percentage of systematic variation. Percentage of spatially correlated and random variation is defined similarly. Following [15], we use 25% systematic, 25% spatially correlated and 50% random variation for all simulations. For greedy image ordering we use $D_1 = 500$ DUTs. For maximum probable estimation error (MPEE) calculation, we use another $D_2 = 500$ DUTs. Both PRS and SET frameworks are evaluated on another set of 1000 DUTs. The highest DUT accuracy is 91.71% for VGG16 and 88.39% for Mobilenet. For evaluating PRS and SET, we fix A_{th} as 5, 10 and 15% lower compared to the maximum DUT accuracy.

The test frameworks are evaluated based on three metrics (1) Test escape (TE) (2) Yield Loss (YL) (3) Effective number of images (N_{eff}). If the accuracy of a DUT satisfies $A < A_{th}$, but the DUT is labeled as "pass" during testing, it is referred as false positive. On the other hand, if a DUT has accuracy $A \geq A_{th}$, but is labeled as "fail" during testing, it is referred as false negative. Test escape is defined as the percentage of false positives with respect to the total number of DUTs. Yield loss

Mobilenet										VGG									
	A_{th} = 83.39			A_{th} = 78.39			A_{th} = 73.39				A_{th} = 86.71			A_{th} = 81.71			A_{th} = 76.71		
m_{prs}	TE	YL	N_{eff}	TE	YL	N_{eff}	TE	YL	N_{eff}	m_{prs}	TE	YL	N_{eff}	FP	YL	N_{eff}	TE	YL	N_{eff}
1.5	2.2	1.2	362	0.7	1.2	279	1	1.4	213	1.5	2.4	2.2	420	1.7	1.2	235	0.7	0.8	269
1.75	1.6	0.7	510	0.4	0.8	418	0.6	0.7	292	1.75	1.2	1.1	592	1.1	0.8	354	0.6	0.4	363
2	0.9	0.6	665	0.4	0.4	517	0.4	0.3	405	2	0.8	0.3	878	0.9	0.5	464	0.3	0.4	453
2.25	0.7	0.3	776	0.3	0.1	647	0.1	0.1	547	2.25	0.6	0.2	1072	0.7	0.2	600	0.2	0.2	530
2.5	0.4	0.1	953	0.3	0	745	0	0.1	642	2.5	0.3	0	1313	0.4	0	725	0.2	0.1	629
2.75	0.1	0	1117	0.1	0	863	0	0	757	2.75	0.1	0	1511	0.3	0	856	0.1	0	718
3	0.1	0	1226	0	0	955	0	0	832	3	0.1	0	1746	0.3	0	982	0.1	0	799

Table I: Progressive Random Sampling: As m_{prs} increases, TE, YL decrease and N_{eff} increases

Mobilenet										VGG									
	A_{th} = 83.39			A_{th}= 78.39			A_{th} = 73.39				A_{th}= 86.71			A_{th} = 81.71			A_{th} = 76.71		
m_{set}	TE	YL	$Neff$	TE	YL	$Neff$	TE	YL	$Neff$	m_{set}	TE	YL	$Neff$	TE	YL	$Neff$	TE	YL	$Neff$
0.6	1.4	1	206	3.2	3	140	4.4	0.9	62	0.6	1.7	2.1	375	10.4	1	4	2.3	4.5	4
0.8	0.3	0.1	486	0	0.2	359	0.8	0.3	219	0.8	0.5	0.2	811	0.4	0.2	314	2.2	0.3	108
1	0	0	707	0	0.1	515	0.1	0	367	1	0	0	1162	0.1	0.2	493	0.1	0	333
1.2	0	0	884	0	0	665	0	0	485	1.2	0	0	1423	0	0	663	0	0	443

Table II: Sequential Estimation Test: As m_{set} increases, TE, YL decrease and N_{eff} increases

is defined as the percentage of false negatives with respect to the total number of DUTs. If the d-th DUT requires N_d images for testing, we define effective number of images as $N_{eff} = \frac{1}{1000}\sum_{d=1}^{1000} N_d$. Any test framework should try to achieve low TE, YL and N_{eff}.

In PRS, m_{prs} is multiplied with "standard deviation of expected error" whereas in SET, m_{set} is multiplied with "maximum probable estimation error". As a result, for reliable testing, we require $m_{prs} > m_{set}$. In our experiments, for PRS, we use $T_{min} = 100$ and we sweep m_{prs} from 1.75 to 3.0. For SET, we sweep m_{set} from 0.6 to 1.2. For greedy image ordering J_{min} is set to 500.

B. Progressive Random Sampling

Table I shows the performance of PRS. In PRS, the range of the confidence interval ($p_{hi} - p_{lo}$) is proportional to the confidence multiplier m_{prs}. As m_{prs} increases, the probability of the true normalized accuracy being within the confidence interval increases. As a result both test escapes and yield loss reduce. On the other hand the termination criterion of PRS requires non-overlap between the normalized accuracy confidence interval ($[p_{lo}, p_{hi}]$) and the normalized accuracy threshold (p_{th}). As a result when m_{prs} increases DUTs require more images for testing and effective number of images (N_{eff}) increases. We observe that for $m_{prs} \leq 2$, there is significant test escape and yield loss. For example, for Mobilenet we observe 0.9% TE and 0.6% YL corresponding to $m_{prs} = 2.0$. For $m_{prs} \geq 2.75$, N_{eff} is high. For example, for VGG16 and $A_{th} = 86.71$, $N_{eff} = 1511$. As a result, to achieve both lower test escapes and yield loss and lower N_{eff} during post-manufacture test, we recommend using $m_{prs} = 2.25$ or 2.5 .

C. Sequential Estimation Test

Table II explains the performance of the SET framework. Similar to PRS, in SET as well, as the confidence multiplier m_{set} increases, TE and YL reduce, but N_{eff} increases. For VGG16 and $m_{set} = 0.6$, TE exceed 10% (for $A_{th} = 81.71$) and YL is 4.5% (for $A_{th} = 76.71\%$) making it unsuitable for post-manufacture testing. For $m_{set} = 1.2$, N_{eff} is 1423 for VGG16 ($A_{th} = 86.71$). Overall, we recommend $m_{set} = 1.0$, which achieves TE and YL less than 0.2% and $N_{eff} \leq 1162$ for all simulation conditions.

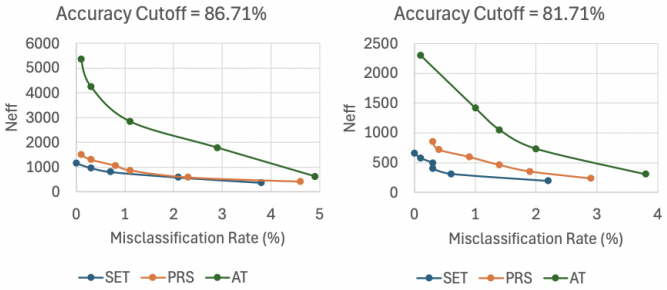

Figure 3: Comparison with progressive random sampling (PRS), sequential estimation test (SET) and alternate test (AT)

D. Comparison with Prior Work

We compare both SET and PRS with alternate test, proposed in [11]. We evaluate the three methods in terms of misclassification rate (sum of TE and YL) and effective number of images (N_{eff}). All three methods aim at achieving low misclassification rate (MR) and low N_{eff}. Figure 3 shows that for any given N_{eff}, SET achieves the lowest MR. Similarly, for any given MR, SET achieves the lowest N_{eff}. For example, for VGG16 and $A_{th} = 86.71$, SET, PRS and AT requires $N_{eff} = 973, 1313$ and 4258 to achieve $MR = 0.3\%$. As a result, for the same MR, SET is $4.4\times$ faster than AT and PRS is $3.2\times$ faster than AT. Similarly, for an accuracy cutoff of 81.71% and MR = 0.3%, SET, PRS and AT requires 407, 856 and 1861 images respectively. It implies that SET is $4.6\times$ faster than AT whereas PRS is $2.2\times$ faster than AT.

VII. Conclusion

This research proposes two novel algorithms for testing compute-in-memory based convolutional neural network accelerators. Progressive random sampling iteratively applies random images from a CNN's testing dataset to characterize a DUT as "pass" or "fail". Sequential estimation test leverages DUT output statistics to rank-order images from the CNN's testing dataset. Both sequential estimation test and progressive random sampling outperform state-of-the-art testing methods in terms of misclassification rate and test speedup.

Acknowledgement

This research was supported by the U.S. National Science Foundation under Grant: 2414361.

REFERENCES

[1] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-memory chips for deep learning: Recent trends and prospects," *IEEE circuits and systems magazine*, vol. 21, no. 3, pp. 31–56, 2021.

[2] S. Yu, X. Sun, X. Peng, and S. Huang, "Compute-in-memory with emerging nonvolatile-memories: Challenges and prospects," in *2020 ieee custom integrated circuits conference (cicc)*. IEEE, 2020, pp. 1–4.

[3] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 14–26.

[4] M. Dazzi, A. Sebastian, L. Benini, and E. Eleftheriou, "Accelerating inference of convolutional neural networks using in-memory computing," *Frontiers in Computational Neuroscience*, vol. 15, p. 674154, 2021.

[5] W. Dobbelaere, R. Vanhooren, W. De Man, K. Matthijs, A. Coyette, B. Esen, and G. Gielen, "Analog fault coverage improvement using final-test dynamic part average testing," in *2016 IEEE International Test Conference (ITC)*, 2016, pp. 1–9.

[6] H.-G. Stratigopoulos and C. Streitwieser, "Adaptive test with test escape estimation for mixed-signal ics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 10, pp. 2125–2138, 2018.

[7] F. Su, C. Liu, and H.-G. Stratigopoulos, "Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives," *IEEE Design Test*, vol. 40, no. 2, pp. 8–58, 2023.

[8] A. Chaudhuri, J. Talukdar, and K. Chakrabarty, "Machine learning for testing machine-learning hardware: A virtuous cycle," in *2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2022, pp. 1–6.

[9] S. Kundu, S. Banerjee, A. Raha, S. Natarajan, and K. Basu, "Toward functional safety of systolic array-based deep learning hardware accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 3, pp. 485–498, 2021.

[10] S. T. Ahmed and M. B. Tahoori, "Compact functional test generation for memristive deep learning implementations using approximate gradient ranking," in *2022 IEEE International Test Conference (ITC)*, 2022, pp. 239–248.

[11] K. Ma, A. Saha, C. Amarnath, and A. Chatterjee, "Efficient low cost alternative testing of analog crossbar arrays for deep neural networks," in *2022 IEEE International Test Conference (ITC)*, 2022, pp. 499–503.

[12] A. Saha, C. Amarnath, K. Ma, and A. Chatterjee, "Signature driven post-manufacture testing and tuning of rram spiking neural networks for yield recovery," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024, pp. 740–745.

[13] A. Fantini, L. Goux, R. Degraeve, D. Wouters, N. Raghavan, G. Kar, A. Belmonte, Y.-Y. Chen, B. Govoreanu, and M. Jurczak, "Intrinsic switching variability in hfo 2 rram," in *2013 5th IEEE International Memory Workshop*. IEEE, 2013, pp. 30–33.

[14] Y. Zhu, G. L. Zhang, T. Wang, B. Li, Y. Shi, T.-Y. Ho, and U. Schlichtmann, "Statistical training for neuromorphic computing using memristor-based crossbars considering process variations and noise," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1590–1593.

[15] A. Saha, K. Ma, C. Amarnath, and A. Chatterjee, "Efficient optimized testing of resistive ram based convolutional neural networks," in *2024 IEEE 30th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2024, pp. 1–7.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[18] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.